Three Encouraging Developments in Software Management©

Esther Derby
Esther Derby Associates, Inc.

When business results aren't what they're supposed to be, companies often make the mistake of trying everything from forced rankings to incentive programs. Instead of these "quick fix" methods, software development managers should consider evidence-based management, as well as Lean and Agile software development techniques, for successful, long-term results.

Twenty-odd years ago, W. Edwards Deming despaired over the state of American management:

> The biggest problem that most any company in the Western world faces is not its competitors, nor the Japanese. The biggest problems are self-inflicted, created right at home by management that are off course. [1]

As I started to write this article, a colleague told me that management in his company is taking decisive action to improve the skill level of employees and increase productivity. This company certainly needs management action to improve business results. They haven't shipped a significant release in over two years, and they've racked up technical debt.

The product development group seems incapable of deciding what they should ship; they change course every few weeks. The result is lots of action, but no completion.

People who are doing their jobs well don't hear from their manager, and neither do those who are failing. Managers complain, but don't clarify expectations, offer feedback, or support people to improve.

The senior managers have decreed that they will raise the overall skill level in the organization and improve productivity. I was intrigued to hear how they planned to accomplish this goal ... until I heard what action they were planning to take. They plan to rank all of the people in the company, with the bottom 5 percent invited to find other employment.

Sometimes, I despair.

But sometimes I dream, because there are moments that make me feel hopeful about management in U.S. companies.

Evidenced-Based Management

Evidence-based management is a movement to look at what actually works in business rather than relying on common practice, fads, and what everyone else happens to be doing.

We've all heard about *known unknowns* and *unknown unknowns*. But Jeffrey Pfeffer and Robert I. Sutton focus on the so-called *knowns* of business practice that *ain't so*, or are only partly so [2].

Pfeffer and Sutton examine the evidence behind the so-called *war for talent*, use of incentive pay, emphasis on strategy, and other widely accepted business practices. In most cases, they find that the data do not support, or only weakly support, the efficacy of these many common business practices.

Take incentive pay, pay-for-performance, and related pay system schemes. Many people in organizations believe that incentives can fix performance problems in organizations. These systems come with cascading goals, objectives, and rating and ranking programs aimed at motivating and rewarding behavior that will benefit the company.

The evidence shows that incentive pay and rating systems overestimate the importance of extrinsic rewards and underestimate the damage incentive pay programs actually do. Pay-for-performance systems not only don't motivate people, they demotivate and often drive behavior the company doesn't want. Yet most companies have some form of incentive pay.

My anecdotal evidence matches the data Pfeffer and Sutton collected. One project manager complained that even though she received a high rating, her annual raise was a pittance—1 percent. On the other hand, a new employee who was still learning the ropes (and was only marginally productive) received a 5 percent raise.

Her manager explained the logic of moving people quickly to the mid-range of the salary base.

While it may sound (sort of) logical, it's the emotional impact that's a problem. "Why should I bust my butt?" the project manager asked. "The best I can hope for is a 1 percent raise. It feels like they don't want me around anymore." Within six months, she'd left the company for a different employer (and a better paying job).

Evidence-based management isn't a quick fix, so I don't expect to see American management transformed overnight. It is a mindset and includes learning and researching, experimenting, and acting on evidence—disciplines that are often at odds with the image of the charismatic, action-oriented executive. As Pfeffer and Sutton say, "Being a 'master of the obvious' may not sound exciting and won't get you labeled as a genius, but it can make and save your company a lot of money" [2].

As a manager, if you're not a master of the obvious, the rest won't much matter.

Evidenced-based management is out there, and many people are reading Pfeffer and Sutton's book or following Sutton's blog. People are starting to ask questions and take management action on the basis of what is known—rather than what they wish was known. I plant the seeds of evidence-based management action wherever and whenever I can, prompting observation, probing for clear thinking, and shoring up hope with facts.

Lean for Software Development

Deming contributed to the post-World War II transformation of Japanese manufacturing. Now, in a sweet turnabout, Americans are examining one of the great success stories of Japanese manufacturing for lessons we can apply in managing software development.

Lean applies key principles that have worked in manufacturing to software development. The Toyota system, from which most Lean thinking derives, has 14 principles. All of them work synergistically, with five that are particularly heartening:

1. **See the system as a whole.** While functional organizations tend to focus on specialized skills and excellence at the component level, Lean teaches managers to see their organizations as systems of interdependent parts. A system view leads to improving the entire system.
2. **Level out the workflow.** Leveling out the workflow involves three things. The first job of management is to reduce overburden. In manufacturing, overburdened machines break down. In knowledge work, overburdened people make

mistakes, fall ill, burn out. The second job is to eliminate unevenness in the workload by figuring out how to create a steady flow. Finally, managers need to eliminate waste. Anything that does not directly add value to a product is considered waste. Table 1 (from [3]) shows the correspondence between the seven manufacturing wastes and the seven software development wastes. Eliminating waste also results in streamlining approval processes, finding the lightest audit requirements that will work, and reorienting budget processes to focus on creating value rather than containing costs.

3. **Create a culture that supports learning and continuous improvement.** The only way organizations improve their results is through feedback, reflection, and problem-solving. One way to support this is retrospectives (a subject near and dear to my heart, and described in [4]). Retrospectives provide a structured way for teams and work groups to examine their technical practices, processes, and collaboration, look for root causes, and plan for improvements. Retrospectives are a plan-do-check-act cycle.
4. **Develop exceptional people and teams.** Investing in people and empowering them leads to better decisions and problem-solving. That means giving people the contextual understanding, skills, and authority to make decisions as close to the work as possible.
5. **Focus on the long-term.** U.S. managers are driven by short-term metrics and financial targets, even when reaching those targets sacrifices long-term results. But to produce exceptional results, managers need to think farther out than the next quarter and cannot cut corners now that will cost the company later.

There is much more to the management philosophy that supports the Toyota Production System[1] and other Lean transformations, but these strike me as the five most critical steps to improving management.

Agile Software Development

Agile methods reinforce key elements of Lean. They aim to manage workflow so that teams work at a sustainable pace—one that they can maintain without burning out. Agile also eliminates waste by focusing only on the most important features, catching defects before they escape an iteration, and reducing documentation to the minimum (and, consequently) allowing the cross-functional development team do their work.

Agile removes the manager from day-to-day task management, and leverages self-organizing teams to create valuable software. The teams manage their own work, and improve their own practices and processes through regular retrospectives. And with the managers freed-up from task supervision, they can turn their talents and energy to removing blocks and impediments that interfere with the team's ability to deliver working software.

The Seven Wastes of Manufacturing	The Seven Wastes of Software Development
Inventory	Partially Done Work
Extra Processing	Extra Processes
Overproduction	Extra Features
Transportation	Task Switching
Waiting	Waiting
Motion	Motion
Defects	Defects

Table 1: *The Seven Wastes of Manufacturing and Software Development*

In many organizations, team's goals cascade down from management. The team works to meet their goals, which in turn contributes to the manager meeting his or her goals. The team is in service to the manager. Agile breaks this dynamic by focusing the team on building the features that have the best return on investment. Teams don't serve their manager, but work towards creating value for the company.

Agile shifts the manager out of task supervision and into enabling productivity. Managers serve the team (and the company) by creating an environment that will enable the team to do their best work and by working across the organization to eliminate waste and impediments.

Conclusion

Clearly the managers at my colleague's company have not yet discovered evidence-based management and have not mastered the obvious. If they had, they'd know that forced ranking won't solve their problems.

They haven't yet realized that their organization is a system, one that is perfectly designed to produce the results they are achieving. They haven't realized that they are punishing workers for management problems, and that improving the skill level of their developers is fruitless without creating an environment where developers can produce results.

Ah, well.

All we can do is plant the seeds, and work with the people who realize their job is not merely telling others what to do, but creating organizations that allow everyone to do their best. And these movements may help us.

References

1. Walton, Mary. Foreword by W. Edwards Deming. The Deming Management Method. Perigree, 1988: xii.
2. Pfeffer, Jeffrey, and Robert I. Sutton. Hard Facts, Dangerous Half-Truths, and Total Nonsense: Profiting from Evidence-Based Management. Boston: Harvard Business School Press, 2006.
3. Poppendieck, Mary, and Tom Poppendieck. Lean Software Development: An Agile Toolkit. Addison Wesley, 2003.
4. Derby, Esther, and Diana Larsen. Agile Retrospectives: Making Good Teams Great. Pragmatic Programmers. Pragmatic Bookshelf, 2006.

Note

1. For more information on the management philosophy behind the Toyota Production System, see "The Toyota Way: 14 Management Principles from the World's Greatest Manufacturer," by Jeffrey Liker.

About the Author

Esther Derby is well known for her work in helping teams grow to new levels of productivity and coaching technical people who are making the transition to leadership roles. She is one of the founders of the Scrum Alliance and co-author of "Agile Retrospectives: Making Good Teams Great." She has a master's degree in organizational leadership and more than two decades experience in the wonderful world of software.

Esther Derby Associates, Inc.
3620 11th AVE
Minneapolis, MN 35401
Phone: (612) 724-8114
E-mail: derby@estherderby.com